PATENT ABSTRACTS OF JAPAN

(11)Publication number : 03-288933
(43)Date of publication of application : 19.12.1991

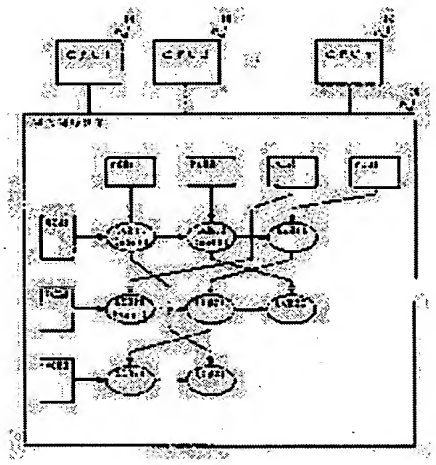
(51)Int.Cl. G06F 9/46

(21)Application number : 02-090314
(22)Date of filing : 06.04.1990

(71)Applicant : HITACHI LTD
(72)Inventor : NITTA ATSUSHI
YONEDA SHIGERU

(54) METHOD FOR CONTROLLING ACCESS TO DATA RESOURCES

(57)Abstract:
PURPOSE: To obtain a high-functional locking mechanism with minimum spin weight by separately providing a locking word to show the possibility of access and a queue control word to control a queue showing the maintaining and waiting states of the lock, and utilizing a forming function in series at a low level.
CONSTITUTION: On a computer system where plural CPUs10 share a main storage device 20, plural process control blocks (PCB) 1 - PCB 4 execute locking requirements LCB 11 - LCB 34 to plural shared resources RCB 1 - RCB 3. In such a case, the locking word showing the state of the lock to the shared resource is separated from the queue control word to control the queue of the locking requirement and the access to the shared resource is efficiently formed in series among the plural processes. Thus, the high-grade locking mechanism can be realized without increasing the load of the CPU by the spin lock so much.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平3−288933

⑤1Int. Cl.⁵	識別記号	庁内整理番号	⑬公開　平成3年(1991)12月19日
G 06 F 9/46	340 H	8120−5B	

審査請求　未請求　請求項の数 6　(全15頁)

⑤4発明の名称　データ資源に対するアクセスを制御する方法

⑳特　願　平2−90314
㉒出　願　平2(1990)4月6日

㉗発明者　新田　淳　神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
㉗発明者　米田　茂　神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
㉑出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地
㉔代理人　弁理士　小川　勝男　外1名

明　細　書

1．発明の名称

データ資源に対するアクセスを制御する方法

2．特許請求の範囲

1．多重プロセス、多重プログラミングシステムにおいて、複数のプロセスによつて共有されるデータ資源へのアクセスを、各プロセスが発行するロック要求を許可するかまたは待たせることによつて制御する方法であつて、共有資源およびロック要求に対応する制御ブロックを設け、該共有資源に対応する制御ブロックは、共有資源のロック状態を示すロックワードと、ロック要求に対応する制御ブロックを含むキューを管理するキューコントロールワードを持ち、該キューコントロールワードはある時点からある時点の間にキューへロック要求に対応する制御ブロックが追加されたことを検出する手段を含み、

(1) ロックワードを参照してロックが許可されるかどうかを判定し、次にロック要求に対応した制御ブロックをキューに追加し、もしロック待ちが必要な場合はロックワードを再び参照して、ロック要求が許可可能な状態に変更されていたらロック待ちを回避するステップと、

(2) キューを走査して許可可能なロック要求を見つけ、新たに許可されたロック要求に対応するようにロックワードを更新し、ロック待ちを解除し、キュー走査開始点からロック待ち解除時点の間に、ロック要求に対応する制御ブロックが、キューに追加されたことを検出したならば、キュー走査から処理を再試行するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

2．多重プロセス、多重プログラミングシステムにおいて、複数のプロセスによつて共有されるデータ資源へのアクセスを、各プロセスが発行するロック要求を許可するかまたは待たせることによつて制御する方法であつて、共有資源およびロック要求に対応する制御ブロックを設け、

該共有資源に対応する制御ブロックは，共有資源のロック状態を示すロックワードと、ロック要求に対応する制御ブロックを含むキューを管理するキューコントロールワードを持ち、該キューコントロールワードは最後にキューに入れられた制御ブロックを指すアンカポインタと、キューに制御ブロックが入れられた回数を記録する追加カウンタとを含み、ロック要求に対応する制御ブロックはキューに１つ先に入れられた制御ブロックを指すポインタと、要求しているロックの種別を示すワードと、ロック要求が待たされていることを示すウエイトフラグを含むことを特徴としたデータ資源に対するアクセスを制御する方法。

３．請求項２においてさらに、

(1) ロックワードを参照してロック要求が許可可能かどうかを判断するステップと、

(2) ロック要求に対応する制御ブロックをキューに追加して、上記追加カウンタを増分するステップと、

(3) 上記キューへの制御ブロックの追加後、ロック待ちが必要であれば、ロックワードを再び参照して、ロックワード内容がロック許可可能に変更されていないかを確認し、もし許可可能であれば、ロック待ちの回避を行なうステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

４．請求項３においてさらに、

(1) キューコントロールワードの追加カウンタを退避するステップと、

(2) キューを走査してロック許可可能な制御ブロックのリストを作成するステップと、

(3) 許可されるべきロックに応じてロックワードを変更するステップと、

(4) 上記 (1)において退避した追加カウンタと現在の追加カウンタを比較し、もし異なっていれば上記 (1)から処理を再試行するステップと、

(5) 上記 (2)で作成された制御ブロックのリスト中のロック待ちを解除するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

５．請求項２においてさらに、キューコントロールワードは、キューに対する専用アクセスの可能性を表す排他フラグと、キューに対する共用アクセスの可能性を表す共用カウンタを含み、ロック要求に対応する制御ブロックは、キュー中の制御ブロックが論理的に削除されたことを表す無効フラグを含み、

(1) 排他フラグがセットされていなければ、共用カウンタを増分してキュー走査を開始し、排他フラグがセットされていればリセットされるのを待つステップと、

(2) キュー走査を行って目的の制御ブロックに無効フラグをセットするステップと、

(3) 共用カウンタを減じた結果が１以上の場合は共用カウンタを減じてキュー走査を終了し、０になる場合は共用カウンタを減じると同時に排他フラグをセットしてキューを走査し、無効フラグが設定された制御ブロックをキューから削除し、排他フラグをリセットしてキュー走査を終了するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

６．請求項１から請求項５のいずれかに示されたデータ資源に対するアクセスを制御する方法を備えたコンピュータシステム。

３．発明の詳細な説明

〔産業上の利用分野〕

本発明は、データ資源に対するアクセスの制御方法、すなわち、電子計算機上のマルチプロセス、マルチプログラミング環境において、並行処理を行なうプロセス間でデータ資源を共有するためにアクセスを並列化する方法に関し、特に、複数の命令プロセッサが主記憶装置を共有するような密結合マルチプロセッサ環境に好適な、共有資源に対するアクセスの制御方法に関する。

〔従来の技術〕

電子計算機上のマルチプロセサス、マルチプロ

グラミング環境では、多数のプロセスが様々な処理を並行して動作できる。プロセスは、オペレーティングシステムがＣＰＵを割当てるための仕事の単位であり、その動作のためには、ＣＰＵ，メモリ，Ｉ／Ｏ等の各種資源を必要とする。ここで、プロセスに固定的に括り付けられている資源の管理については、何等の問題も生じない。しかし、複数のプロセスによって共有される資源に関しては、それらの資源に対するアクセスの競合を制御し、直列化する何らかの方法が必要である。

複数のプロセスによるアクセス競合を直列化する最も低レベルの方法は、一般にリード・モディファイ・ライト型命令と呼ばれる機械命令によつて与えられる。この命令は、主記憶上に格納された状態変数を参照し、その値にしたがつて状態変数を書き換える動作を、１つの原子的動作として実行する。例えば、ＨＩＴＡＣ　Ｍシリーズ処理装置（日立製作所製電子計算機）におけるＣＳ(Compare and Swap)命令、およびＣＤＳ(Compare Double and Swap)命令が、このような命令である。これらの機械命令の詳細は、(株)日立製作所発行の「ＨＩＴＡＣ　Ｍシリーズ処理装置」マニュアルに記載されている。

リード・モディファイ・ライト型の命令においては、その命令を実行するＣＰＵが上記状態変数を参照してから更新するまでの間、その状態変数を含む主記憶領域をロックして、他ＣＰＵによる当該主記憶領域へのアクセスをウエイトさせるという制御を、ハードウエアによつて実現している。

ソフトウエアレイヤにおけるアクセスの直列化機構は、上記リード・モディファイ・ライト型命令を用いて実現される。マドニック（Ｓ.Ｅ. Madnick)およびドノバン（Ｊ.Ｊ.Donovan)による“オペレーティング　システムズ（Operating Systems)”（１９７４月刊，McGraw－Hill社)の４．５章に示されているように、このような機構は典型的には、「ロック」および「アンロック」と呼ばれる命令系列からなる。

この機構においては、ある共有資源が使用中であるかどうかを表示するために、ロックワードと呼ばれる資源対応の状態変数が設けられる。

ロック動作では、ロック要求プロセスは、ロックワードを参照して、もし使用中でなければロックワードを使用中に変更し、もし使用中であれば、再びロック動作を試行する（このロック動作の繰り返しはスピンウエイトと呼ばれる）。アンロック動作においては、プロセスは、ロックワードを使用中から非使用中に変更する。この方法では、ロックを確保しているプロセスは、共有資源に対して排他的な使用権を有し、他のプロセスはロック保有プロセスがアンロックを行なうまで、共有資源にアクセスできない。

ロック動作は、ロックワードを参照・更新するために、リード・モディファイ・ライト型命令を使用する。これは、あるプロセスが、ロックワードが使用中でないと判定してからロックワードを使用中に変更する処理の間に、他プロセスがロックワードを書き換えてしまうことを防ぐためである。

このようなロック機構は、しばしばスピンタイプのロックあるいは簡単にスピンロックと呼ばれる。

ソフトウエアによるロック機構には、上記スピンロックの他に、サスペンドタイプのロック（もしくはサスペンドロック）と呼ばれるものがある。サスペンドロックのロック動作においては、ロックが即時に確保できない場合、ロック要求プロセスは先入れ先出しのキューに入れられ、サスペンド状態となる。すなわち、ロック要求プロセスは、ロック待ちの間に、スピンウエイト処理によってＣＰＵを占有することなく、ＣＰＵを他の実行可能なプロセスに明け渡す。アンロック動作実行プロセスは、新たにロック確保が可能となるプロセスがキュー中に存在すれば、そのプロセスをキューから取り出してサスペンド状態を解除する。サスペンドロックは、スピンロックに比べて、CPUを有効に利用することが可能であるが、キュー操作とプロセスの状態変更処理の余分な動作を必要とする。また、一般にキューは、主記憶上に設けられたプロセス対応の制御ブロックをポインタチ

エーンで結ぶことによつて実現されるが、このポインタチエーン自体が、ロツク動作およびアンロツク動作を並行して行なう複数のプロセスによつてアクセスされる共有資源であり、アクセスの直列化を必要とする。このキユー操作のアクセスの直列化は、スピンロツクによつて実現されることが多かつた。

特開昭60－128537には、スピンロツクに依存することなく、リード・モデイフアイ・ライト型命令を直接使用して、サスペンドロツクを実現する方法が示されている。この方法においては、ロツクワードは、アクセスの可能性を示すロツクフラグと、ロツク待ちのプロセスを表現するキユーのアンカとなるロツクポインタからなり、ロツクワード全体をリード・モデイフアイ・ライト型の命令で参照・更新することによつて、排他および共有の２つのモードを備えたロツク機構を実現している。

〔発明が解決しようとする課題〕

上記の単純なスピンロツク方法では、ロツク確保要求の競合が発生した場合、ロツクを確保できた、ただ１つのプロセスを実行するＣＰＵを除いては、ＣＰＵがスピンウエイト処理に使用され、本来の業務処理が中断される。これは、特に、複数の命令プロセツサが主記憶装置を共用するような、いわゆる密結合マルチプロセツサ環境においては、無視できない業務処理性能の低下を引き起こす。スピンロツクは、オペレーテイングシステムの最も基本的で単純な処理の直列化に使われるべきものであり、より高レベルの複雑な処理の逐次化には向いていない。

サスペンドロツク方法は、スピンロツクより処理が複雑であるが、ロツク待ちのプロセスがCPUを占有することがないため、高レベル処理の逐次化に適している。しかし、サスペンドロツク処理要求が頻発するようになると、ロツク待ちのキユー操作を逐次化するためのスピンロツクによる性能劣化が顕著になつてくる。

上記特開昭60－128537に開示された方法を用いると、スピンロツクを用いることなく、共用および排他の２つのロツクモードを備えたサスペンドタイプのロツクを実現でき、高レベル処理の逐次化を効率よく行なうことができる。しかし、単なる共用・排他の２モードのロツクだけでは、ソフトウエア階層のより上位の処理における、複雑な逐次化機能の要求に対応できない。１つのロツクワード内に、共有資源の占有状態を表示するロツクフラグと、ロツク待ちを管理するキユーのアンカポインタを設け、ロツクフラグとロツクポインタをまとめてリード・モデイフアイ・ライト型命令によつて参照・更新することが、実現可能なロツク機能に対する制限をもたらす。これは、ロツクワードの長さに制約が存在するためである。

典型的なハードウエアで実現されているリード・モデイフアイ・ライト型命令は、倍語長以下の主記憶領域を操作するものであり、通常アンカポインタは１語を占有するため、ロツクフラグは１語長以内で表現されなければならない。これは、多種多様なロツクモードを表現する場合に、大きな制約となる。

またこの方法では、ロツク待ちを管理するキユーの操作において、キユーの最後の１つまたは連続する複数の要素を、リード・モデイフアイ・ライト型命令を用いて、要素のネクストポインタを更新することにより削除している。この方法は、キユー要素の削除を行なうプロセスが、同時にはただ１つしか存在しないという前提のもとでだけ正しく動作する。共用・排他の２モードのロツクでは、キユー要素の削除は、ロツクモードの変わり目となるアンロツク処理時、すなわち、排他ロツク解放、または最後の共用ロツクの解放時にだけ実行されるため、上記前提条件が満たされる。しかし、より複雑なロツク機能では、このような前提条件は必ずしも満足されず、複数のアンロツク処理実行プロセスがキユーを同時に操作するタイミングが発生しうる。もちろん、この２モードロツク方法を、内部的な制御ロツク操作の逐次化に利用することにより、より高機能なロツク処理プログラムを作成することが可能である。しかし、低レベルの機能（すなわちリード・モデイフ

アイ・ライト型命令）を直接利用して、高機能なロック方法を実現できるならば、その方が望ましい。

本発明は、以下のような背景のもとに考案されたものであり、その目的は、高レベルの機能を備えたソフトウエアロック機構を、できるだけ低レベルの直列化機能を利用して、効率的に実現することである。

〔課題を解決するための手段〕

本発明の目的は、アクセス可能性を示すロックワードと、ロックの保有および待ち状態を示すキューを管理するキューコントロールワードを分離して設け、キューコントロールワードは、キューへの要素の追加をカウントする追加カウンタと、最後にキューに入れられた要素を指すキューアンカポインタを含むようにして、ロックワードおよびキューコントロールワードを以下のように参照・更新することによって実現できる；

(1) ロック処理は、ロックワードを参照してロック確保が即時に可能か、それとも待たなければならないかを判断し、ロック要求をキューに登録し、もしロックを待つのであれば、キュー登録後、ロックワードを参照してロック確保が可能かどうか再度確認し、ロック確保可能であればアンロック処理の一部を実行してロック待ちを回避する。

(2) アンロック処理は、はずキューコントロールワードの追加カウンタを記憶し、ロック許可処理を実行し、追加カウンタの値を、先に記憶しておいたカウンタ値と比較して、もし変更されていたら、アンロック処理を再試行する。

(3) ロック許可処理は、キューを走査してロック許可が可能な待ち状態のロック要求を確保し、ロックワードを新しいロック状態に更新し、ロック待ちを解除する。

また、本発明の目的は、キューコントロールエリアに、キューアクセスに対する排他・共用ロックを示すキューロックフラグを設け、また、キュー要素に削除された要素であることを示す削除フラグを設け、以下のようにキュー操作を行なうことによって実現される；

(1) キューへの要素の追加は、キューアンカポインタへ、追加要素のアドレスを設定し、追加カウンタをカウントアップすることによって行なう。

(2) キューからの要素の削除では、キューに対する共用ロックを確保して、キューを走査し、削除すべき要素に削除フラグを設定して、キューに対する共用ロックを解放する。この時、キューに対する共用ロックを最後に解放するプロセスが、共用ロックの解放と同時にキューに対する排他ロックを確保し、削除フラグが設定されている要素を実際にキューから削除する。キューに対して排他ロックが設定されている場合、キューに対する共用ロック要求はスピンウエイトする。

〔作用〕

共有資源に対するロックの状態を示すロックワードと、ロック要求のキューを管理するキューコントロールワードを分離したことによって、複雑なロック機能を実現することが容易になる。すなわち、ロックワードおよびキューコントロールワードはそれぞれ、本発明が適用されるコンピュータシステムのハードウエアによって規定される、リード・モディファイ・ライト型命令が扱える最大のオペランド長まで占有できるようになる。

一方、ロックワードとキューコントロールワードを分離して、別々に更新するため、ロックワードの状態とキューの状態との整合性をとるための機構が必要となる。ロック処理時の、キューイング後のロックワード再確認と、アンロック処理時の、キューコントロールワード中の追加カウンタの記憶および確認が、このような機構を与える。ロック処理時のロックワード再確認は、ロックワードを参照してから、キューイングするまでの間に、アンロック処理によって、ロックワードが変更されたことを検出する。また、アンロック処理時の追加カウンタの操作は、キューを走査してロック許可可能なロック要求を確定している間に、新たな待ち状態のロック要求がキューに追加されたことを検出する。これによって、ロック処理とアンロック処理が競合して、ロック許可可能であるにもかかわらず、待ち状態が解除されないロッ

ク要求が発生してしまうタイミングを救済できる。

一方、キューコントロールワードの排他・共用ロック方法は、非常に少ないスピンウエイトで、一般的なキュー操作を行なうことを可能とする。キューへの要素の追加は、キューコントロールワードを、直接リード・モディファイ・ライト型命令で操作することによつて実行され、キュー要素への削除フラグの設定は、キューに対する共用ロックを確保して実行される。これらの処理は、複数のプロセスによつて同時に実行可能である。特に、アンロック処理で複数のプロセスがロック許可処理のためにキューを同時にアクセスすることが可能となる。共用ロックを最後に解放するプロセスは、共用ロックの解放と同時に、キューに対する排他ロックを確保し、キュー中の排他フラグが設定されている要素をキューから実際にはずして、排他ロックを解放する。この、排他ロック占有の間に、他プロセスがアンロック処理を実行しようとして、キューをアクセスする場合はスピンウエイトするが、このような競合が発生する確率は十分に低いことが期待できる。

以上のように、本発明によれば、最小限のスピンウエイトの上に、高機能なロック機構を実現することができる。

〔実施例〕

以下、本発明の実施例を図に基づいて詳細に説明する。ここでは、２つの実施例を述べる。第１の実施例においては、全てのロック要求、すなわち、許可されているロック要求と、待たされているロック要求の双方が、キューに保持される。第２の実施例においては、待たされているロック要求だけで、キューに保持される。

(1) 第１の実施例

（１－１）コントロールブロックの構成

第１図は、本発明の第１の実施例の概要を示すコンピュータシステムのブロック図である。第１図では、プロセスが共有資源に対してロックを要求して許可されている、または待たされている、状態が、主記憶（もしくは仮想記憶）上の制御ブロック群と、それらの間のポインタチエーンによつて表現されている。共有資源はリソースコントロールブロック（ＲＣＢ）１００によつて、プロセスはプロセスコントロールブロック（ＰＣＢ）１１０によつて、ロック要求はロックコントロールブロック（ＬＣＢ）１２０によつて、それぞれ示される。共有資源１００に対するロック要求１２０は、ポインタチエーン１４０によつて、最新に発せられたロック要求から順につながれる。また、プロセス１１０が行つたロック要求１２０は、ポインタチエーン１５０によつてつながれる。すなわち、ロック要求１２０が、どのプロセスからの、どの共有資源に対する要求であるかということが、２種類のポインタチエーン１４０および１５０によつて一意的に特定される。

第２図に、本発明が実施されるコンピュータシステムの環境と、制御ブロック間のポインタチエーンの実例を示す。第２図では、複数のＣＰＵ１０（ＣＰＵ１～ＣＰＵｎ）が、主記憶装置２０を共有しているコンピュータシステム上で、複数のプロセスＰＣＢ１～ＰＣＢ４が、複数の共有資源ＲＣＢ～ＲＣＢ３に対してロック要求ＬＣＢ１１～ＬＣＢ３４を行つている様子が示されている。本発明は、このような、複数のＣＰＵが主記憶装置を共有する密結合マルチプロセッサシステムでのマルチプログラミング環境において、複数のプロセス間の共有資源へのアクセスを効率的に直列化する方法を与える。本発明は、また、CPUが１台であるようなコンピュータシステムでのマルチプログラミング環境において、複数プロセスの共有資源へのアクセスを直列化する場合にも、有効である。

第２図に示すポインタチエーンのうち、横方向のチエーンは、特定の共有資源に対するロック要求のキューを表す。例えば、ＲＣＢ１からＬＣＢ１１，ＬＣＢ１２，ＬＣＢ１４に延びるポインタチエーンは、ＲＣＢ１によつて代表される共有資源に対するロック要求を、要求発生時間の新しい順につなぐものである。ＬＣＢ中に（wait）表示のあるものは、そのロック要求が現在待ち状態であることを示す。第２図では、ＲＣＢ１に対して

最も古くに発行されたロック要求ＬＣＢ１４は許可されたが、その後に発行されたロック要求LCB１２とＬＣＢ１１は、即時にロックが許可されず、ロック要求ＬＣＢ１４が解放されるのを待つている状態が示されている。ＲＣＢ２およびＲＣＢ３から右方向に延びるポインタチエーンについても同様である。

これに対し、縦方向のポインタチエーンは、特定のプロセスが発行したロック要求のリストを表す。例えば、ＰＣＢ１からＬＣＢ１１，ＬＣＢ３１へと延びるポインタチエーンは、ＰＣＢ１によつて示されるプロセスが、ロック要求ＬＣＢ１１，ＬＣＢ３１を、これとは逆順に発行した事を示す。ロック要求３１は許可されたが、ロック要求１１は許可されず、プロセスＰＣＢ１は、現在待ち状態である。ＰＣＢ２，ＰＣＢ３，ＰＣＢ４から下方向に延びるポインタチエーンも同様である。１つのプロセスは、０以上の任意の数だけ許可されたロック要求を待つことができるが、待ちのロック要求は高々１つしか待てない。

ここで再び第１図にもどつて、各制御ブロックの詳細を説明する。

共有資源を代表するリソースコントロールブロック（ＲＣＢ）１００は、共有資源に対するロックの状態を示すロックワードＬＷと、共有資源に対するロック要求のキューを管理するキューコントロールワードＱＣＷを含む。ここで、本発明の１つの特徴は、ＬＷとＱＣＷが隣接している必要がないことである。ロックワードＬＷの内容の詳細については、後述する。キューコントロールワードＱＣＷは、排他フラグＸ，共有カウンタSHR，削除フラグＤ，追加カウンタＡＤＤ，キューアンカポインタＡＮＣからなる。排他フラグＸと共用カウンタＳＨＲは、複数プロセスによつて同時にアンロック処理が行われる場合の、キューに対するアクセスの直列化を制御するために用いられる。削除フラグＤは、キュー中に、ポインタチエーンにはつながつているが、論理的には削除されているＬＣＢが存在することを示すために用いられる。追加カウンタＡＤＤは、ＬＣＢが新たにキューに入れられるごとにカウントアップされ、ロック処理とアンロック処理の競合が発生したことを検出するために用いられる。キューアンカポインタＡＮＣは、ロック要求のキューを表すポインタチエーン１４０の始点である。

プロセスを代表するプロセスコントロールブロック（ＰＣＢ）１１０は、そのプロセスに対して許可されている、または、許可されるのを待つているロック要求のリストを表すポインタチエーン１５０の始点となるポインタＬＰＴを含む。

ロック要求を表すロックコントロールブロック（ＬＣＢ）１２０は、ＲＣＢからのポインタチエーン１４０における次のＬＣＢを示すポインタＮＩＱと、ＰＣＢからのポインタチエーン１５０における次のＬＣＢを示すポインタＮＩＳと、要求しているロックのモードを示すＭＯＤＥと、フラグＦＬＧを含む。フラグＦＬＧは、ＬＣＢの状態を示すものであり、ロック要求が待ち状態であることを意味するＷフラグ、ＬＣＢがキューすなわちＲＣＢからのポインタチエーン１４０内で、無効であることを意味するＩフラグ，ロック要求が許可可能となつて待ち状態を解除する必要があることを意味するＰフラグを含む。ＬＣＢは、また、待ち状態を解除する必要があるロック要求をつなぐポインタチエーン１６０における次のLCBを示すポインタＮＩＰを含む。

ロック処理／アンロック処理は、以上のコントロールブロックの他に、いくつかのワークレジスタ１３０を使用する。ワークレジスタの１つは、待ち状態解除を行なうべきロック要求をつなぐポインタチエーン１６０の始点となるポインタPPTとして使われる。もう一つのワークレジスタは、アンロック処理中で、新たに許可可能となつたロック要求のモードを記憶するためのΔＭＯＤＥとして使われる。

第３図は、ロックワードＬＷの内容を示している。本実施例におけるロックは、ＥＵ／ＥＲ／ＰＵ／ＰＲ／ＳＵ／ＳＲの６種類のモードを持つ。第４図は、これらのロックモード間の両立性を示す。第４図のマトリクス中で○印は２つのモード

が同時に許可可能であることを、×印は、２つのモードが背反であることをそれぞれ示している。このマトリクスからわかるように、ＥＵ／ＥＲ／ＰＵモードのロックは、ただ１つのプロセスにだけ許可され、ＰＲ／ＳＵ／ＳＲモードのロックは同時に複数のプロセスに許可されうる。このため、第３図のロックワードＬＷ中では、ＥＵ／ＥＲ／ＰＵモードは、各々フラグによって、ＰＲ／ＳＵ／ＳＲモードは各々カウンタによって表現される。ロックワードＬＷは、また、許可を待っているロック要求の数を示すカウンタＷＡＩＴを含む。１つでも待ちのロック要求が存在する場合、その後に到着したロック要求は全て待ちとなる。これは、ロック要求の時間順序性を保証するためである。ロックモードの種類、およびロックモードのロックワードＬＷ中における表現方法については、第３図および第４図に示したものに限らず、任意のものが可能である。ここでのただ１つの制限は、ロックワードＬＷが、リード・モディファイ・ライト型の命令によって、更新可能であるということである。

（１－２）ＣＳおよびＣＤＳ命令

ロック処理／アンロック処理の詳細に入る前に、そこで使用されるリード・モディファイ・ライト型機械命令であるＣＳ(Compare and Swap)命令とＣＤＳ(Compare Double and Swap) 命令について簡単に説明しておく。

ＣＳ命令は、２つの汎用レジスタと主記憶（仮想記憶）上の１語長の領域をオペランドとし、第１のレジスタと主記憶（仮想記憶）上の１語の内容を比較して、もし両者が等しければ、その１語を第２のレジスタの内容で書き換えて条件コードを０に設定し、もし両者が異なれば、第１のレジスタにその１語の内容をロードして条件コードを１に設定する動作を、原子的な１動作として実行する、リード・モディファイ・ライト型の命令である。

ＣＤＳ命令も、ＣＳ命令と同様の動作を実行するが、こちらは、４つの汎用レジスタを用いて倍語長の領域を操作するところが異なる。複数のＣＰＵが主記憶を共有する密結合マルチプロセッサでは、あるＣＰＵが実行中のプロセスが主記憶上の状態変数の内容を比較し、その値に従ってその状態変数を変更しようとする場合に、通常のロード／コンペア／ストア命令を使用したのでは、状態変数を読み込んで（ロード）、比較して（コンペア）、書き換える（ストア）処理の間に、状態変数が他のＣＰＵによって書き換えられ、処理の整合性が失われてしまうことがありうる。このためＣＳ命令やＣＤＳ命令を使用して、比較と書き換えを原子的な１動作で行うようにするのである。

以下で、ロック処理／アンロック処理を、流れ図に基づいて詳細に説明する。なお、説明が不必要に複雑になることを避けるため、図では細かな初期設定処理や例外処理を省いてある。

（１－３）ロック処理

第５図は、ロック処理の流れ図である。ロックを要求するプロセスは、入力情報として、ＲＣＢアドレス，ＰＣＢアドレス，要求ロックモードを指定して、ロック処理をコールする。ロック処理では、まずＬＣＢを作成して、その内容を初期化し、ＰＣＢからのポインタチェーンにつなぐ（ステップ１００１）。この初期化では、ＮＩＱおよびＮＩＰは空値に、フラグは全てリセット状態に、ＭＯＤＥは要求ロックモードに設定される。また、ＬＣＢをＰＣＢからのポインタチェーンにつなぐため、ＮＩＳには旧ＬＰＴの値が、新ＬＰＴとしてＬＣＢアドレスが、それぞれセットされる。

次に、ロックワードＬＷを参照して、要求したロックが許可可能か、待たなければならないかを判定し（ステップ１００２）、ＣＤＳ命令を用いて新しいロックワードをセットする（ステップ１００３）。待ち状態のロック要求が存在せず、かつ、要求したロックモードが既に他のプロセスに許可されているロックモードと両立可能な場合にだけロックが許可される。新たなロックワードは、ロック許可可能な場合は、要求ロックモードに対応するフラグをセットする，もしくはカウンタを上げることにより、ロック待ちの場合は、ＷＡＩＴカウンタを上げることによって得られる。

ステップ１００３のＣＤＳ命令が失敗するのは、他プロセスのロック処理（ステップ１００３）もしくはアンロック処理（後述の第７図ステップ１２０７）によるロックワードＬＷの更新とのアクセス競合が発生したためであり、ステップ1002から再試行する（ステップ１００４）。ロック待ちが必要であればＬＣＢのＷフラグをセットする（ステップ１００５，ステップ１００６）。

続いて、ＣＤＳ命令を用いて、ＬＣＢをキューに入れる（ステップ１００７）。この処理は、現在のＡＮＣの値をＮＩＱにコピーした後、ＣＤＳ命令を用いて、ＱＣＷの内容が他プロセスによって変更されていないことを確かめながら、ＡＮＣの値をＬＣＢアドレスに書き換え、かつＡＤＤカウンタの値を１だけ増加させることによって実行される。ＡＤＤカウンタが最大値である場合には、１へラップアラウンドする。ステップ１００７のＣＤＳ命令が失敗するのは、他プロセスのロック処理（ステップ１００７）またはアンロック処理（後述の第６図ステップ１１０７、第８図ステップ１３０２，１３０４）によるＱＣＷの更新と競合が発生したためであり、キューイング処理を再試行する（ステップ１００８）。

ＬＣＢをキューに入れた後、ロックが許可されるならばロック処理コール元にリターンし、ロック待ちが必要ならばモードの再確認を行なう（ステップ１００９）。モード再確認処理では、再びロックワードを参照してロックが許可可能になっているかどうか確かめ（ステップ１０１０、ステップ１０１１）、ロック許可がやはり不可能である場合はロック要求プロセスを待ち状態にする（ステップ１０１２）。もし、ロックワードが許可可能に変更されている場合は、ロック待ちを回避するためにアンロック２処理をコールする（ステップ１０１３）。このようなモード再確認を行うのは、ステップ１００３でロックワードをセットしてから、ステップ１００７でＬＣＢをキューにつなぐまでの間に、現在ロックを許可されているプロセスがアンロック処理を実行してロックワードＬＷが、ロック許可可能な状態に書き換えられた時、今回のロック要求が待ちのままでキューに残る事を防ぐためである。ＬＣＢがキューにつながれていれば、アンロック処理がキューを走査して待ちのＬＣＢを見つけることができ、ロック待ちは解除される。ロック処理でロックワードを参照する以前に、アンロック処理でのロックワード更新が行われていれば、ロック要求は待ちにならない。

問題は、競合するアンロック処理がロックワードを更新する以前に、ロック処理でロックワードを参照し、かつ、ロック処理でＬＣＢをキューにつなぐ以前にアンロック処理でキューを走査する、というタイミングで２つの処理が実行された場合である。この場合、アンロック処理によって、このＬＣＢの待ちが解除されることは期待できないため、ＬＣＢのキュー登録後にロック処理内でモードを再確認し、必要であればアンロック処理を模擬するのである。

従来の方法では、ロックワードＬＷとキューコントロールワードＱＣＷは、一体化されており、１回のＣＤＳ命令で同時に更新されていたため、このようなタイミングの狭間が発生することはなかった。本発明では、ロックの自由度を増大するために、ロックワードＬＷとキューコントロールワードＱＣＷを分離しており、そのために、モード再確認処理が必要となったのである。

（１－４）アンロック処理

第６図～第９図に、アンロック処理の流れを示す。アンロック処理は、２つのエントリポイントを持つ。ロックを許可されたプロセスが、そのロックを解放する場合にコールするのがアンロック１エントリポイントである。これに対し、前述したロック処理のモード再確認処理(ステップ1013)においてコールされるのがアンロック２エントリポイントである。入力情報は、双方のエントリポイントともＬＣＢアドレスである。

第６図は、アンロック処理の初期部の流れである。通常のアンロック処理、すなわちアンロック１エントリポイントから始まる処理では、まずワークレジスタＡＭＯＤＥに、ロック解放によって

発生するロックワードＬＷの変更分、すなわちＬＣＢのＭＯＤＥの逆数をセットする（ステップ１１０１）。次にＣＳ命令を用いてＬＣＢのＩフラグをセットする（ステップ１１０２）。このＣＳ命令は、Ｉフラグのセットが成功するまで繰り返される(ステップ１１０３)。続いて、ＬＣＢをＰＣＢからのポインタチエーンよりはずす（ステップ１１０４）。すなわち、このＬＣＢがポインタチエーンの先頭であればＬＰＴに、そうでなければこのＬＣＢの１つ前のＬＣＢのＮＩＳに、このＬＣＢのＮＩＳの値をコピーする。以上がアンロック１エントリポイントに個有の前処理であり、以後の処理はアンロック２エントリポイントと共通である。アンロック２エントリポイントから処理が開始された場合、ワークレジスタΔMODEの初期値は０である。

次に、ロックワードＬＷ、およびキューコントロールワードＱＣＷ内のＡＤＤカウンタの値を、空きのワードレジスタまたはワークエリアに退避し（ステップ１１０５）、Ｘフラグがセットされているかを見る（ステップ１１０６）。Ｘフラグがセットされていることは、他プロセスがキューを占有してアクセスを行っていることを意味しており、Ｘフラグがリセットされるまでスピンウエイトする。Ｘフラグがリセットされていることを確認したら、ＣＤＳ命令でキューコントロールワードＱＣＷのＳＨＲをカウントアップする（ステップ１１０７）。このＣＤＳ命令が成功した場合、アンロック処理実行プロセスは、キューに対する共有アクセス権限を得たことになる。ＣＤＳ命令が失敗するのは、他プロセスによるロック処理またはアンロック処理とのＱＣＷ更新における競合が発生したためであり、ステップ１１０５から再試行する（ステップ１１０８）。ここまでが、アンロック処理の第１段階である。

第７図は、許可が可能となったロック要求をキュー中で見けて、待ちを解除する処理の流れである。この処理は、キューコントロールワードQCWのＡＮＣポインタからポインタチエーンをたどり、最も古くに待ち状態となったＬＣＢを捜すことから始まる（ステップ１２０１）。すなわち、Ｗフラグがセットされ、かつ、ＩフラグとＰフラグがリセットされているような、ポインタチエーン中の最も後ろのＬＣＢを見つける。許可可能なロック要求がもはや存在しない場合、このような状態のＬＣＢは見つからず、ループ終了処理に移る（ステップ１２０２）。

待ちを解除すべきＬＣＢが見つかつたならば、ＣＳ命令でＬＣＢにＰフラグをセットする（ステップ１２０３）。ＣＳ命令が成功すれば（ステップ１２０４）、ロックワードの差分情報をワークレジスタΔＭＯＤＥに加え（ステップ１２０５）、ＬＣＢをワークレジスタＰＰＴからのポインタチエーンにつなぐ（ステップ１２０６）。ΔMODEの変更では、ＷＡＩＴカウンタを１減じ、ＬＣＢが要求するロックモードに対応するフラグのセット、もしくはカウンタの増分を行なう。また、ＬＣＢのＮＩＰに現在のＰＰＴの値をコピーし、ＰＰＴにこのＬＣＢのアドレスをセットすることによつて、ＬＣＢをＰＰＴからのポインタチエーンにつなぐことができる。ステップ１２０３のＣＳ命令が失敗するのは、ＬＣＢのフラグＦＬＧに対するアクセスの競合が起きた場合であり、ステップ１２０１から処理を再試行する。

ステップ１２０６までの処理が完了したら、待ちを解除すべき次のＬＣＢを捜すため、ステップ１２０１にもどる。このようにして、ΔＭＯＤＥには、ロックワードを変更するための差分情報が累積され、ＰＰＴからのポインタチエーンには、待ちを解除すべきＬＣＢのリストが形成される。

ＬＣＢサーチのループから抜けると、ΔMODEに蓄積されたモード差分情報を用いて新しいロックワード内容を作り、ＣＤＳ命令でロックワードＬＷにセットする（ステップ１２０７）。この場合、ＣＤＳ命令でＬＷと比較するのは、ステップ１１０５で退避しておいたロックワードの内容である。このＣＤＳ命令が失敗するのは、ステップ１１０５からステップ１２０７までの間に他プロセスがＬＷを更新したためであり、再び、ループの先頭（ステップ１２０１）にもどつて、待ちが

解除できるＬＣＢがないか確かめる（ステップ１２０８）。ＣＤＳ命令が成功したならば、PPIからのポインタチエーンにつながれているＬＣＢの待ちを全て解除し（ステップ１２０９）、第８図のアンロック終了処理に制御を移す。

第８図は、アンロック処理の終了部の流れである。ここでは、ステップ１１０７で確保したキューに対する共用アクセス権限の解放処理を行なう。まず、ＱＣＷのＳＨＲカウンタの内容を見て（ステップ１３０１）、キューにアクセスしている他プロセスが存在する、すなわちＳＨＲの値が１よりも大きい場合は、ＣＤＳ命令を用いてＳＨＲの値を減じる（ステップ１３０２）。もし、アンロック処理１エントリポイントから処理が開始されているのであったならば、このＣＤＳ命令で同時にＤフラグをセットする。ＣＤＳ命令が成功すればアンロック処理を終了してコールもとへリターンする（ステップ１３０３）。一方、キューをアクセスしているプロセスが自プロセスだけの場合、ＳＨＲの値は１である。この場合は、ＣＤＳ命令を用いてＳＨＲの値を減じると同時にＸフラグをセットする（ステップ１３０４）。もし、アンロック処理１エントリポイントから処理が開始されている場合には、やはり、Ｄフラグを同時にセットする。ステップ１３０４のＣＤＳ命令が成功してＱＣＷにＸフラグをセットするということは、キューに対する排他アクセス権限を確保できたことを意味する。この場合、第９図のＬＣＢ削除処理に制御を移す（ステップ１３０５）。

ステップ１３０２もしくはステップ１３０４のＣＤＳ命令が失敗するのは、他プロセスによるロック処理またはアンロック処理とのＱＣＷアクセスの競合が発生した場合である。どちらの処理と競合したかを確認するために、ＡＤＤの値が、ステップ１１０５で退避した値と等しいかどうかを見る（ステップ１３０６）。ＡＤＤの値が変わっていなければ、他プロセスのアンロック処理との競合であり、ステップ１３０１からＱＣＷの更新処理を再試行すればよい。ＡＤＤの値が変わっていれば、他プロセスによるロック処理と競合が発生したためであり、ステップ１２０１からキュー走査処理全体をもう一度行う必要がある。ロック処理によってＡＤＤが更新されたということは、新たなＬＣＢがキューに追加されたことを意味する。このＬＣＢが、ロック許可可能な、待ち状態のＬＣＢではないかどうかを確認するために、キュー走査を再試行するのである。このようなLCB状態が発生するのは、まずロック処理がロックワードＬＷを参照して待ちを決定し(ステップ1002)、次にアンロック処理がキューを走査して待ちを解除可能なＬＣＢを捜し（ステップ１２０１～ステップ１２０６）、その後にロック処理がＬＣＢをキューに追加してプロセスを待ち状態にし（ステップ１００７～ステップ１０１２）、最後にアンロック処理が上記待ちロック要求を許可可能とするようなモードにロックワードを更新する（ステップ１２０７）、というタイミングで２つの処理が同時実行される場合である。ロック処理がLCBをキューに追加するよりも、アンロック処理がロックワードを更新する方が先であれば、ロック処理のモード再確認処理（ステップ１０１０）が、競合発生を検出してＬＣＢの待ちを解除する。このように、ロック処理におけるモード再確認動作と、アンロック処理におけるＡＤＤカウンタ確認動作の双方を備えることによって、ロック要求が不要に待たされるタイミングが発生することを防止できる。

最後に、第９図にしたがって、ＬＣＢのキューからの削除処理を説明する。まず、ＱＣＷのＤフラグを参照して、キュー中に削除すべきＬＣＢが存在するか判定する（ステップ１４０１）。Ｄフラグがセットされていなければ、Ｘフラグをリセットしてアンロック処理を終了する（ステップ１４１２）。Ｄフラグがセットされている場合には、ＡＮＣからポインタチエーンをたどって最初のＬＣＢを得る(ステップ１４０２)。最初のLCBのＩフラグがセットされていれば、ＣＳ命令でそのＬＣＢをキューからはずして（ステップ1404、ステップ１４０９）、ＬＣＢをメモリの空き領域として解放する（ステップ１４１１）。先頭の

ＬＣＢをキューからはずすのは、ＬＣＢのＮＩＱの値をＡＮＣに設定することによつて行う。ステップ１４０９のＣＳ命令が失敗するのは、他プロセスのＱＣＷ更新処理との競合した場合であり、ステップ１４０２から処理を再試行する（ステップ１４１０）。ステップ１４１１でＬＣＢを解放した場合も、次のＬＣＢを得るためにステップ１４０２にもどる。このようにして、ＡＮＣの指す先頭のＬＣＢが存在しなくなつた場合は、ループを抜ける（ステップ１４０３）。

先頭のＬＣＢのＩフラグがセットされていない場合は、ＮＩＱをたどつて次のＬＣＢを得（ステップ１４０５）、そのＬＣＢのＩフラグがセットされていたら（ステップ１４０７）、ＬＣＢをキューからはずして解放する（ステップ１４０８）。この場合、１つ前のＬＣＢのＮＩＱに、はずすべきＬＣＢのＮＩＱをコピーすることによつて、キューからの削除が実行される。ここでのＮＩＱの更新にＣＳ命令を用いる必要はない。このようにして、キューの最後のＬＣＢまで処理が進んだらループから抜ける（ステップ１４０６）。

アンロック処理では、キュー中の任意の位置のＬＣＢが削除および解放される可能性があるため、キューのポインタ操作を行なうプロセスを、ただ１つに限定する必要がある。ＬＣＢの削除処理はＱＣＷのＸフラグをセットして実行するため、他プロセスのアンロック処理によるキュー操作と競合してポインタチエーンが不正になることはない。他プロセスのアンロック処理は、Ｘフラグがリセットされるまでスピンウエイトする（ステップ１１０５～ステップ１１０６）。Ｘフラグは、ＳＨＲを１から０に変更するプロセス、すなわち、並行してキューアクセスを行つていた複数のプロセスのうちで最後にキューアクセスを終了するプロセスによつてだけセットされ、Ｉフラグがセットされたキューからの削除が終ると直ちにリセットされる。したがつて、スピンウエイトが発生する確立は非常に小さいものである。一方、ロック処理によるキューアクセスの競合は、ステップ１４０９のＣＳ命令で検出する。ロック処理では、キューに対するアクセスがＡＮＣへの新ＬＣＢの追加に限定されているため、スピンウエイトを用いないでアンロック処理キュー操作と処理を直列化することができるのである。

(2) 第２の実施例

第１０図は、本発明の第２の実施例の概要を示すコンピュータシステムのブロック図である。第２の実施例が第１の実施例と異なる点は、第２の実施例においては、ＬＣＢは、もはやあるプロセスがロックを許可されている状態を示す役目を持たずに、ロック待ち表すためだけに用いられることである。したがつて、各プロセスには高々１個のＬＣＢしか対応せず、ＰＣＢ、およびＰＣＢからＬＣＢをつなぐポインタチエーンは不要となる。ＲＣＢやワークレジスタに関しては、第１の実施例と同様である。

第２の実施例におけるロック処理およびアンロック処理は、キューにＬＣＢを登録するのが、ロック待ちの場合だけであることを除いては、第１の実施例とほぼ同様であり、ここで詳しく述べることはしない。第５図において、ステップ1001を削除し、ステップ１００５でＹＥＳの場合は即時にリターンし、ステップ１００６でＬＣＢを作成するように処理を変更すれば、第２の実施例におけるロック処理の流れ図が得られる。同様に、第６図において、全てのアンロック処理をアンロック２エントリポイントから開始するようにし、第７図において、ステップ１２０３でＬＣＢのＰフラグと同時にＤフラグもセットするように処理の流れを変更すれば、第２の実施例におけるアンロック処理の流れ図が得られる。

〔発明の効果〕

以上のように、本発明によれば、複雑なモードを備えたロック機構を、リード・モディファイ・ライト型命令と最小限のスピンロックだけによつて効率的に実現することができる。特に、複数のＣＰＵが主記憶を共有するような、密結合マルチプロセッサ環境において、スピンロックによるＣＰＵの負荷増大をあまり発生させずに、高度なロック機能を実現できる効果がある。

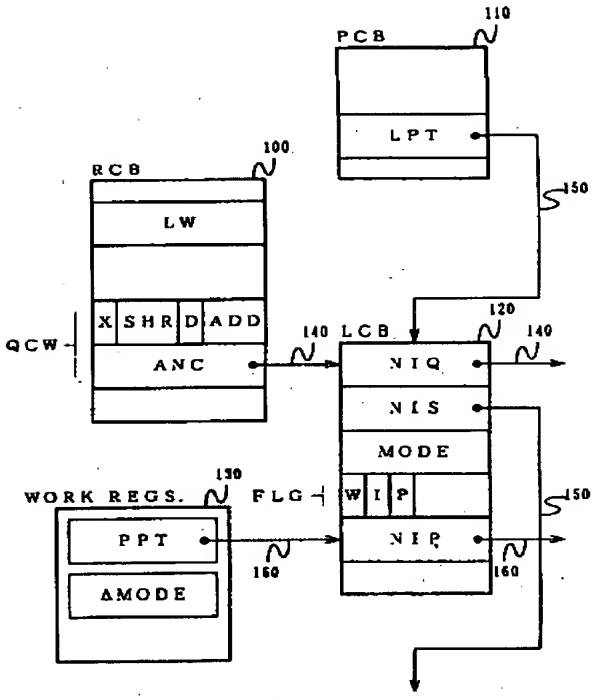
4．図面の簡単な説明

第１図は本発明の第１の実施例を示すコンピュータシステムのブロック図、第２図は本発明が有効に働く計算機環境と主記憶内容を示したブロック図、第３図はロックワードの詳細の説明図、第４図はロックワードの両立性マトリクスの説明図、第５図はロック処理の流れ図、第６図から第９図はアンロック処理の流れ図、第１０図は本発明の第２の実施例を示すコンピュータシステムのブロック図である。

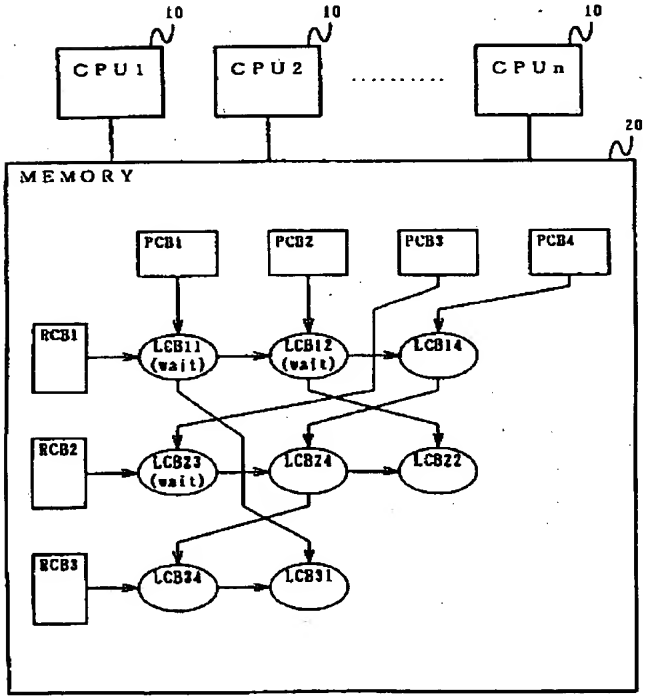
１０…ＣＰＵ、２０…主記憶（仮想記憶）装置、
１００…リソースコントロールブロック（ＲＣＢ）、
１１０…プロセスコントロールブロック（ＰＣＢ）、
１２０…ロックコントロールブロック（ＬＣＢ）、
１３０…ワークレジスタ、１４０…ＡＮＣポインタチエーン、１５０…ＬＰＴポインタチエーン、
１６０…ＰＰＴポインタチエーン。

代理人　弁理士　小川勝男

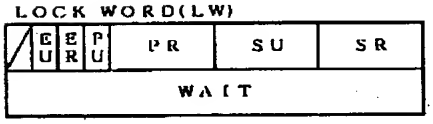
第１図



第２図



第３図



第４図

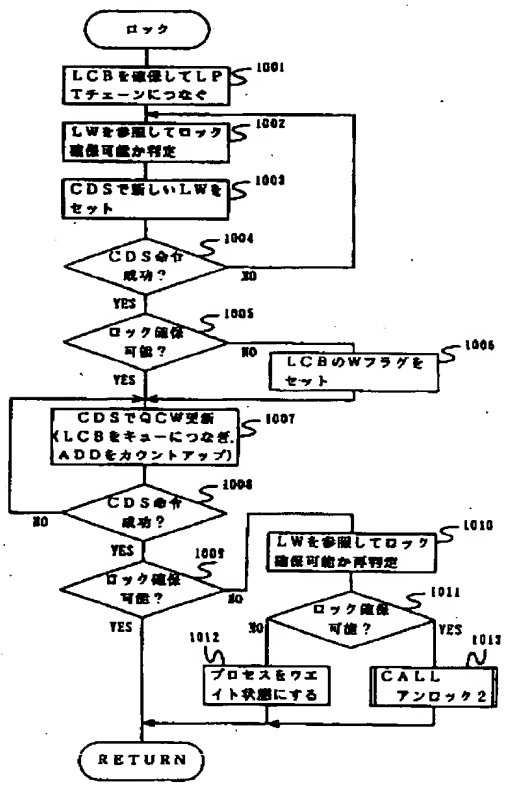
ロック両立性マトリクス

保有されたロックモード ＼ 要求されたロックモード	SR	SU	PR	PU	ER	EU
SR	○	○	○	○	×	×
SU	○	○	×	×	×	×
PR	○	×	○	×	×	×
PU	○	×	×	×	×	×
ER	×	×	×	×	×	×
EU	×	×	×	×	×	×

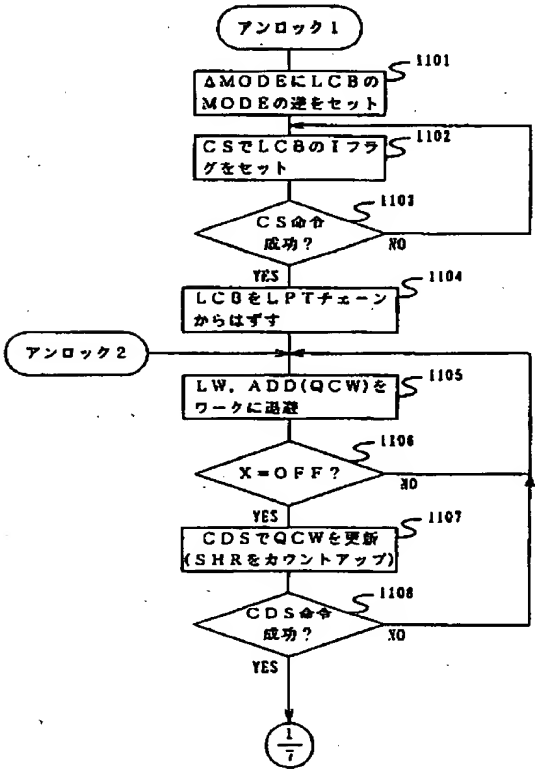
○ ： 両立する
× ： 両立しない

(1)モードの１文字目＝他プロセスに許可する処理
S(Shared) ： リード／ライトを許す
P(Protected) ： リードは許すがライトは許さない
E(Exclusive) ： リード／ライトを許さない
(2)モードの２文字目＝自プロセスで行なう処理
R(Retrieve) ： リードを行なう
U(Update) ： リード／ライトを行なう

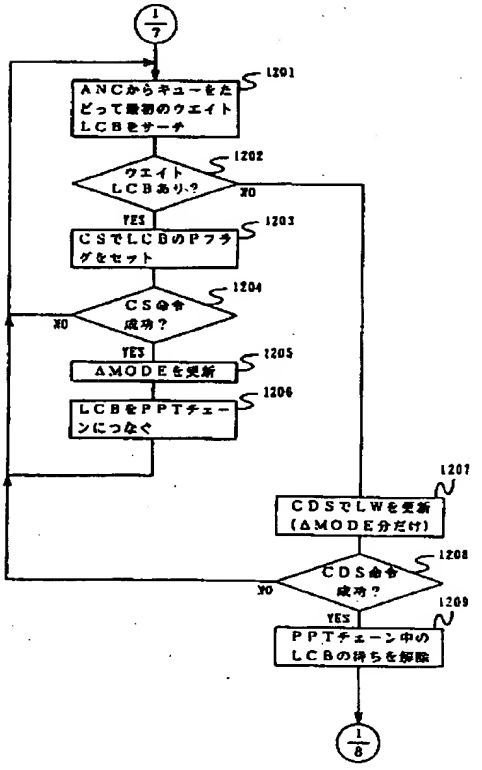
第５図



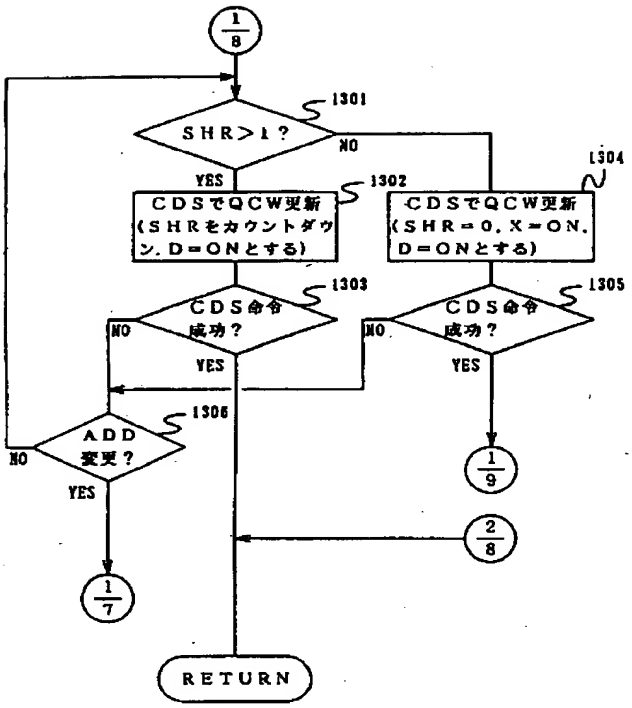
第６図



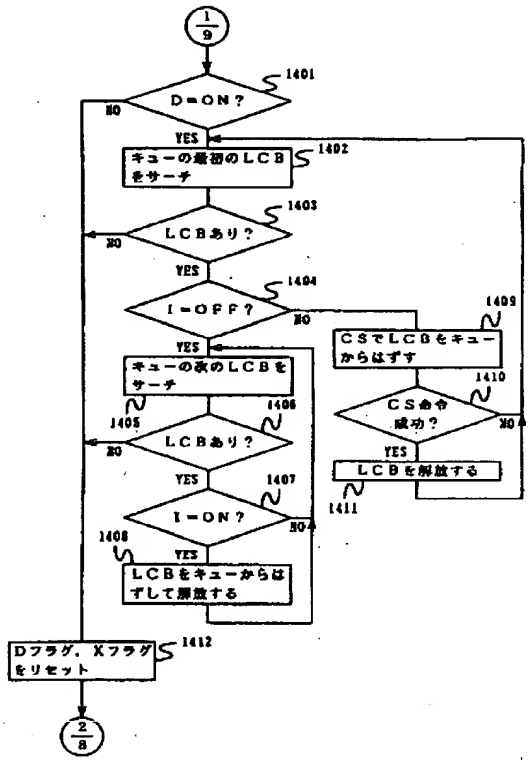
第７図



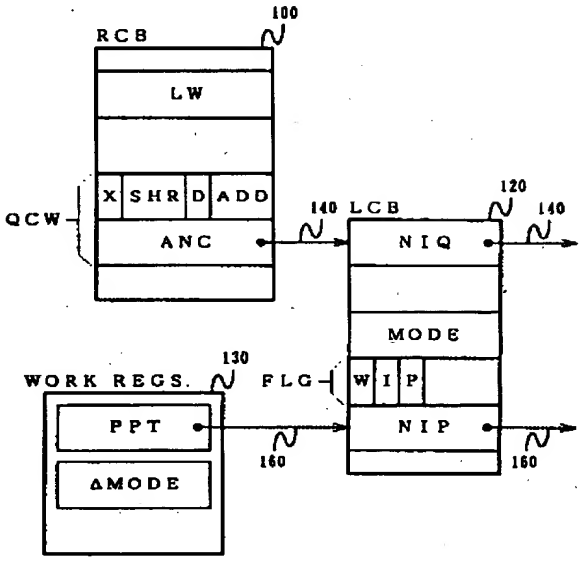
第８図


1/8
1301
SHR>1?
NO
YES
1302
CDSでQCW更新
(SHRをカウントダウン. D=ONとする)
1304
CDSでQCW更新
(SHR=0, X=ON, D=ONとする)
1303
CDS命令
成功?
NO
YES
1305
CDS命令
成功?
NO
YES
1306
ADD
変更?
NO
YES
1/9
2/8
1/7
RETURN

第９図


1/9
1401
D=ON?
NO
YES
1402
キューの最初のLCBをサーチ
1403
LCBあり?
NO
YES
1404
I=OFF?
NO
YES
1409
CSでLCBをキューからはずす
1410
CS命令
成功?
NO
YES
LCBを解放する
1411
キューの次のLCBをサーチ
1406
1405
LCBあり?
NO
YES
1407
I=ON?
NO
YES
1408
LCBをキューからはずして解放する
1412
Dフラグ, Xフラグをリセット
2/8

第１０図


RCB
100
LW
X
SHR
D
ADD
QCW
ANC
140
LCB
120
140
NIQ
MODE
130
WORK REGS.
FLG
W
I
P
PPT
NIP
160
160
ΔMODE